Clustering Composite SaaS Components in Cloud Computing using a Grouping Genetic Algorithm

Zeratul Izzah Mohd Yusoh and Maolin Tang, *Senior Member*, *IEEE*
Science and Engineering Faculty
Queensland University of Technology
Brisbane, Australia
{z.mohdyusoh, m.tang}@qut.edu.au

***Abstract*—Recently, Software as a Service (SaaS) in Cloud computing, has become more and more significant among software users and providers. To offer a SaaS with flexible functions at a low cost, SaaS providers have focused on the decomposition of the SaaS functionalities, or known as composite SaaS. This approach has introduced new challenges in SaaS resource management in data centres. One of the challenges is managing the resources allocated to the composite SaaS. Due to the dynamic environment of a Cloud data centre, resources that have been initially allocated to SaaS components may be overloaded or wasted. As such, reconfiguration for the components' placement is triggered to maintain the performance of the composite SaaS. However, existing approaches often ignore the communication or dependencies between SaaS components in their implementation. In a composite SaaS, it is important to include these elements, as they will directly affect the performance of the SaaS. This paper will propose a Grouping Genetic Algorithm (GGA) for multiple composite SaaS application component clustering in Cloud computing that will address this gap. To the best of our knowledge, this is the first attempt to handle multiple composite SaaS reconfiguration placement in a dynamic Cloud environment. The experimental results demonstrate the feasibility and the scalability of the GGA.**

***Index Terms*—Cloud Computing, Composite SaaS, Clustering, Grouping Genetic Algorithm.**

I. Introduction

Cloud computing [1] is an emerging computing paradigm in which applications, data and IT resources are provided as a service to users over the Internet. One kind of service that can be offered through the Cloud is Software as a Service or SaaS [2]. Nowadays, SaaS is receiving considerable attention from software vendors as well as software users. A report from the International Data Corporation (IDC) states that there will be a significant increase in companies' subscriptions for SaaS practices in their company in the near future [3]. In fact, within three years, companies that have provided SaaS could generate up to an 18 percent increase in revenue, as reported in Dubey&Wagle [4]. Not only that, advances in Cloud computing have provided an efficient means for SaaS hosting; and, therefore, have made SaaS more accessible to a wide range of software users. All these echo the fact that SaaS has become more and more significant among software users and providers.

Recently, SaaS providers have focused on developing SaaS that would be able to effectively address different levels of user's functionalities. One of the ways to achieve this is through decomposition of the software, or composite SaaS. A composite SaaS is a group of loosely coupled individual applications that communicate with each other in order to form a higher-level functional system or application [5]. Through this way, providers can gain a number of benefits including reduced delivery cost, flexible offers of the SaaS functions and decreased cost of subscription for users. However, this approach also introduces new challenges for SaaS resource management in a data centre.

Large-scale data centres like Cloud data centres usually consist of thousands of physical servers with network links. There are also storage servers that are located within the data centre. Virtualization technology is used to achieve simultaneous use of resources in the physical servers as well as lowering the cost for clients. Through virtualization technology, a single physical server is sliced into a number of virtual machines (VMs) where each of the VMs represents an isolated execution environment in the Cloud. These VMs are assigned a chunk of their physical servers' resources including processing capacity, memory and secondary storage. The VMs can have their own applications and operating systems.

At the initial stage of the SaaS deployment process, the SaaS application components and their data components are placed onto the physical servers and storage servers. The application components are then deployed at the virtual machine for execution. The virtual machine that hosts the application components must have sufficient resources in order to fulfil the performance level of certain applications, as specified in terms of the client's Service Level Agreement (SLA). Due to the dynamic environment of a Cloud data centre, where the workload of applications and resource capacities keep changing over time, the initial deployment may need to be modified. As such, the scheduled reconfiguration of the VM is triggered at a certain period of time to maintain the performance of the composite SaaS as well as to minimize the resource usage. In order to do this, the current application component placement needs to be re-configured. One of the ways to achieve this is by clustering two or more application components into a VM. The placement reconfiguration in existing resource management for Cloud data centres often ignores the communication or dependencies between the ap-

plication components in their implementation. In a composite SaaS, these elements are important to be included, as they will directly affect the performance of the SaaS. This paper will propose a solution for multiple composite SaaS application component clustering in the Cloud that will address this gap.

The remainder of the paper is organized as follows. Section II discusses related work. The problem formulation is described in Section III. Section IV presents the proposed algorithm. Then Section V is about the evaluation that has been carried out. The concluding remarks are presented in Section VI.

II. Related Work

Recently, resource management for Cloud data centres has been actively studied and large parts of the work fall into optimizing the resource management in the data centre. The common objectives for optimization include minimizing the resource usage while maintaining the application's performance [6], [7], [8], [9], minimizing the data centre's power consumption [10], [11] and balancing the thermal distribution among the servers [12]. These objectives are achieved through various management plans at different levels. For instance, at the platform level, most existing works focus on the management of VM mapping to physical servers, while at the application level, the plan is to manage VM resources based on the application's workload.

Existing works on resource management at the platform level apply migration of the VM as the main method of dealing with dynamic changes in the Cloud environment [7], [9], [13]. The VM migration method is used as it allows better utilization of resources at the physical servers. Authors in [9] proposed a two-phase solution for VM reconfiguration in a data centre, named Entropy. In the first phase, the minimum number of physical servers that can host the current VMs is determined. In this phase the problem is formulated as the Constraint Satisfaction Problem (CSP), where the constraints are the capacities of the servers. . The second phase of the problem concerns finding the cheapest reconfiguration plan of the VM, based on the physical servers found in the first phase. The cost is determined by the migration's overhead that occurs during the reconfiguration process, where the overhead is calculated based on the memory requirement of the virtual machine that is being migrated. The solution in this work is triggered by the current status of the VM. Other work like the one proposed in [7] triggers the migration periodically, based on its maintenance schedule. They proposed an algorithm that consists of four main processes, which are selection of the physical server that needs migration, selection of the suitable VM on that physical server, selection of the new physical server and assignment of the VM to the physical server. The selection is based on load profiles as well as the behaviour of the servers. All these works at the platform level consider a VM as an independent entity where it does not need to communicate with other VM or storage servers in completing its task. This paper proposes a different approach that concerns the communication involved between VMs and it will be tackled at the application level.

The communication among VMs is highlighted in [6] where the authors proposed a solution for reconfiguration placement that supports three types of constraints which are the VMs demands, communications and availability. The data centre is modelled as a hierarchical structure that represents communication costs based on its hierarchy. Another work that also concerns the communication among VM is presented in [10]. In this paper, they consider a multi-tier application where the deployment may span over multiple VMs. The proposed solution is designed at two levels. At the application level, there is a controller that will dynamically assign resources to applications based on their requirement, and at the platform level, they propose a consolidation algorithm to re-map VMs to physical servers in the case of overload problems. The aim at the platform level is to optimize the data centre's power usage. A similar work can be found in [11] where a multi-level solution is also proposed. The authors in this paper highlight the implementation of the adaptive technique at the application-level, where the application adapts automatically to the availability of the resources, and at the resource-allocation level where the resources allocated adapts to the dynamic workload requirements. There is another level considered in this work, where the power consumption is adapted to the demands at the resource-power level. Our work differs from all these solutions in the sense that they do not consider a composite application, in which a VM can host multiple components with different requirements. In addition, the components have to work with other components to achieve the overall applications' functionalities that are subject to the user's SLA. This paper will propose a solution to address this gap.

III. Problem Formulation

As mentioned in the Introduction, composite SaaS applications and data components are placed in the Cloud's physical machine and storage servers during the initial phase, and later are executed in virtual machines. Fig. 1 illustrates a high level of such a scenario, where the different shapes represent an application component of a composite SaaS, and a VM can host multiple components at a time. In order to deliver a higher level of functionality to users, a composite SaaS may span over multiple VMs.

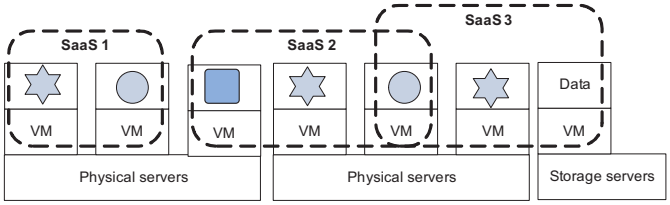

Figure 1. An example of multiple composite SaaS placement in a Cloud data centre.

Due to the dynamic environment of a data centre, resources that have been initially allocated to SaaS applications'

components may be overloaded or wasted. A typical data centre usually schedules a placement reconfiguration where this activity occurs at certain periods of time based on its need. Different approaches can be taken at different periods of time, and can be done either dynamically or statically. In order to obtain an optimal solution, our approach is to deal with the dynamic environment at a static point of time, where a whole data centre will be considered.

The problem of clustering multiple composite SaaS components is aiming to reconfigure the initial placement by clustering the components such that the new placement can minimize the resources used while satisfying the SaaS SLA. The problem's inputs are:

- A Cloud's data centre with its physical servers and storage servers. The physical servers may consist of at least one virtual machine.
- The Cloud's data centre network topology with its links between physical servers and storage servers.
- Multiple composite SaaS with their resource requirements and constraints according to their SLA, and the current placement of the components in the Cloud data centre.

A. Cloud Data Centre Modelling

A Cloud data centre consists of physical servers and storage servers. Each server has its own resource capacities including processing capacity, memory size and storage capacity. Each physical server has at least one virtual machine (VM), where the VM is given slices of the resources capacity of a physical server. A value is assigned to every VM, which will represent the 'cost' of the VM. Each resource type is given a value, and the VM cost is determined based on the capacity of the resources that the VM has. Table 1 summarizes the data centre attributes.

Table I
SETS AND ATTRIBUTES OF CLOUD RESOURCES

Cloud resources	Description
$cs_x \in CS$	The x^{th} physical server, cs_x, in CS, where CS is a set of k physical servers and $1 \leq x \leq k$
$ss_i \in SS$	The i^{th} storage server, ss_i, in SS, where SS is a set of r storage servers and $1 \leq i \leq r$
$vm_{x,y} \in VM$	The y^{th} virtual machine, vm, for cs_x and VM is a set of all virtual machine, $y \leq \mathbb{N}$
$PC_{vm_{x,y}}$	Processing capacity for $vm_{x,y}$
$MC_{vm_{x,y}}$	Memory capacity for $vm_{x,y}$
$ST_{vm_{x,y}}$	Secondary storage for $vm_{x,y}$
$C_{vm_{x,y}}$	Cost of $vm_{x,y}$

B. Cloud Network Topology

The Cloud network is represented by an undirected graph where $G = \langle V, E \rangle$. $V = \{CS \cup SS\}$ is the sets of vertices including physical servers and storage servers, $e \in E$ is the set of undirected edges connecting the vertices, if and only if there exists a physical link transmitting information from v_i to v_j, where $v_i, v_j \in V$. $B_{v_i,v_j} : E \rightarrow \mathbb{R}^+$and $L_{v_i,v_j} : E \rightarrow \mathbb{R}^+$is the bandwidth and latency functions of the link from v_i to v_j respectively.

C. Composite SaaS Modelling

As mentioned earlier, there are multiple composite SaaS deployed in a Cloud data centre at a time. Each of the composite SaaS has its own application and data components with its minimum requirement for resources, as well as its SLA. In this paper, we will consider the maximum response time of the SaaS only as the SLA attribute. The SaaS modelling presented here is made general enough to represent a composite SaaS. Table II summarizes the SaaS components' requirements, and their workflow.

Table II
SETS, PARAMETERS AND REQUIREMENTS OF COMPOSITE SAAS

SaaS modelling	Description
$SC_i \subseteq S$	The i^{th} composite SaaS, SC_i in S. S is a set of n composite SaaS, SC , and $1 \leq i \leq n$
$ac_{i,j} \in AC$	The j^{th} application component, $ac_{i,j}$ for SC_i and AC is a set of all application component, $1 \leq j \leq z$
$dc_{i,q} \in DC$	The q^{th} data component, $dc_{i,q}$ for SC_i and DC is a set of all data component, $1 \leq q \leq x$
$wf_{i,p} \in WF$	A p^{th} business workflow for SC_i where $WF \subseteq AC$, $1 \leq p \leq y$
rt_{SC_i}	The maximum response time for SC_i
$TS_{ac_{i,j}}$	Task size of $ac_{i,j}$
$M_{ac_{i,j}}$	Memory requirement of $ac_{i,j}$
$SZ_{ac_{i,j}}$	Size of $ac_{i,j}$
$AD_{ac_{i,j}}$	Amount of read/write task of $ac_{i,j}$
$W_{wf_{i,p}}$	Weighing for $wf_{i,p}$

Apart from the attributes defined in Table II, there are also other inputs and constraints concerning the current placement. These are defined as:

- A current placement configuration, P, of application components AC, onto virtual machines, VM, given as $P : AC \rightarrow VM$ where $ac_{i,j} \mapsto P(ac_{i,j}) = vm_{x,y}$.
- A current location, L, of the data components, DC, at storage servers, SS, given as $L : DC \rightarrow SS$ where $dc_{i,q} \mapsto L(dc_{i,q}) = ss_k$.

D. Problem's Constraints

There are four types of constraints of the problem as follows:

1) Resource Constraints: For all application components placed in a virtual machine, the total requirements of the resources must not exceed the VM's capacity. This is expressed by the equations below for processing capacity, memory and secondary storage, respectively:

$$\forall_{vm_{x,y} \in VM} \sum_{ac_{i,j} \in AC} TS_{ac_{i,j}} \leq PC_{vm_{x,y}} \mid P(ac_{i,j}) = vm_{x,y} \quad (1)$$

$$\forall_{vm_{x,y} \in VM} \sum_{ac_{i,j} \in AC} M_{ac_{i,j}} \leq MC_{vm_{x,y}} \mid P(ac_{i,j}) = vm_{x,y} \quad (2)$$

$$\forall_{vm_{x,y} \in VM} \sum_{ac_{i,j} \in AC} SZ_{ac_{i,j}} \leq ST_{vm_{x,y}} \mid P(ac_{i,j}) = vm_{x,y} \quad (3)$$

2) Placement Constraints: There are two types of placement constraint: a) An anti-location constraint that determines the list of virtual machines that should not be considered for hosting a specific component, $ac_{i,j}$. The list is defined as $AL = \left\{(ac_{i,j}, vm_{x,y})_z, ...\right\}$ where $z \in \mathbb{N}$, b) An anti-colocation constraint that determines the list of application components that cannot be placed in the same virtual machine. The list is defined as $ACL = \left\{(ac_{i,j}, ac_{s,t})_w, ...\right\}$where $w \in \mathbb{N}$. The solution must comply with the anti-location and anti-colocation constraint defined in the lists:

$$ac_{i,j} \mapsto P(ac_{i,j}) \neq vm_{x,y}, \forall (ac_{i,j}, vm_{x,y}) \in AL \quad (4)$$

$$P(ac_{i,j}) \neq P(ac_{p,q}), \forall (ac_{i,j}, ac_{p.q}) \in ACL \quad (5)$$

3) Response time constraints: The total execution time of a composite SaaS is calculated based on four numerical attributes: a) the time taken for transferring data between the storage servers and the virtual machine, b) the processing time of a component in a selected virtual machine, c) the execution time of a path in the SaaS workflow, and d) the sum of the execution time of the critical path of each workflow multiplied by its weighting. All these attributes have been defined in our previous work [17]. Based on these four values, the total execution time of the SaaS, TET, is determined. The TET must not exceed the maximum response time of a SaaS as agreed in the users' SLA. This constraint is defined as below:

$$TET(SC_i) \leq r_{SC_i} \quad (6)$$

4) Sequence of migration constraints: : To change the placement from one virtual machine to another, the solution has to consider the sequence of components that need to be moved based on the current placement at that time. This sequence may affect the cost of changing the placement directly. Two scenarios will be considered in this problem:

- Sequential move: A particular component can only be moved when another one has been completed. This is in the case where the migrations of two components cannot be done in parallel because of insufficient resources (processing capacity/memory/secondary storage) in the destination virtual machine for one of the components. This is because the virtual machine contains another component that is due to be migrated. As such, the latter component needs to be moved first to free some resources for the other component.
- Cyclic move: The migration of a set of components may need an intermediate destination machine. This is the case where two or more components need to exchange places. This can create a cyclic constraint if the machines involved have insufficient resources.

Given all the input defined above, the objective of the problem is to find a new placement of S onto VM by clustering the application components AC, such that the placement will minimize the resources' costs while satisfying the SaaS constraints. As component placement reconfiguration is an expensive process, the proposed solution will try to achieve the objective with a minimum number of changes to the current placement configuration.

IV. The Proposed Solution Design

From the computational point of view, this problem is a large-scale and complex combinatorial optimization problem with constraints, for which an evolutionary computation technique would be suitable. As the approach for this problem is to cluster components into VMs, the Grouping Genetic Algorithm (GGA) technique is a natural choice. GGA [15] is a modified version of Genetic Algorithm (GA) [16] where it is designed for solving grouping problems. While the GA treats its chromosomes and cost function as a whole, the GGA divides its chromosomes based on relevant groups and the optimization of the cost functions is done based on the grouping. The genetic operations in the GGA are also done based on the defined groups. This is to ensure that the groups can fully explore the search space in order to find the optimal solution.

In the following we discuss the design of the GGA in detail.

A. Chromosome Representation

The chromosome is grouped based on composite SaaS in the Cloud. Each group has two compartments. The first compartment contains n genes, each of which corresponds to an application component in that particular group. The second compartment contains the ID of the VM, where the application component would be placed in the new placement plan. Fig. 2 shows an instance of the chromosome representation, where the total number of composite SaaS is q, and each of the SaaS has a different number of application components.

B. Infeasible Solutions

The chromosomes generated in a solution may be infeasible due to constraints that a SaaS implies. There are four types of constraint defined in Section III that need to be satisfied by each of the chromosomes. The first three constraints concern the SaaS requirements and the maximum response time. All the solutions that do not comply with these constraints will be repaired. The repairing technique performs a simple check in each group to find any combinations that violate the

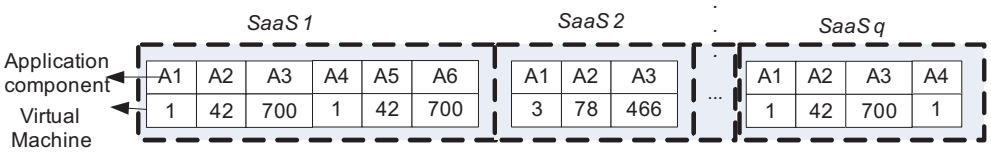

Figure 2. An example of GGA chromosome encoding scheme with q composite SaaS with different number of application components

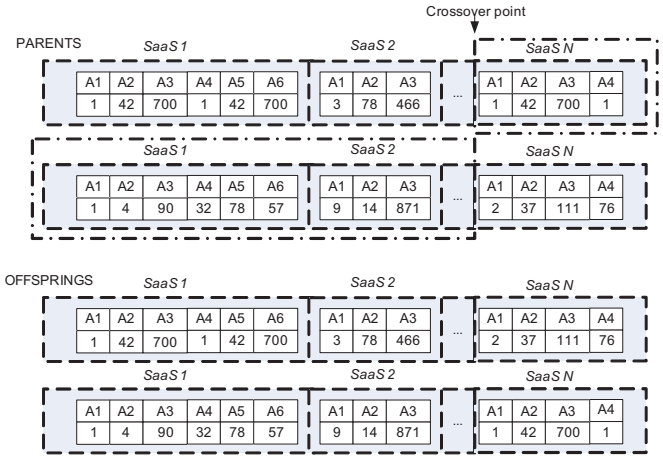

Figure 3. An example of inter-group crossover operation among composite SaaS

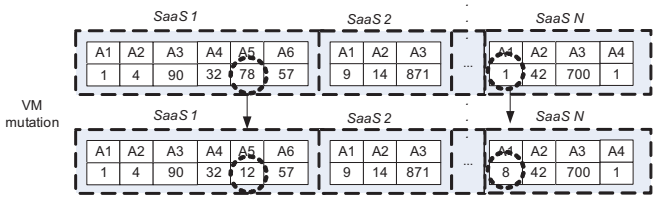

Figure 4. An example of inner-group mutation operation within a SaaS

constraints. A new value will be generated randomly to replace the invalid one. The fourth constraint concerns the sequence of migration, which will affect the migration's cost. This constraint will be incorporated in the fitness function that is described in Section IV-D.

C. Genetic Operators

1) Crossover: The crossover operation is design based on the grouping chromosomes. A single point inter-group crossover will be used. This will combine segments from different SaaS, and produce two offsprings. The top two fittest among the parents and children are selected for the next generation. Fig. 3 illustrates the crossover operation.

2) Mutation: To promote further exploration in the search space, an inner-group mutation operator is used in order to keep the diversity of chromosomes in the population. The mutation operator is applied within a composite SaaS. It changes a VM for a component to another VM that also satisfies all the constraints. Fig. 4 shows an example of the mutation operation.

D. Fitness Function

The aim of the problem is to create groups of components of the multiple composite SaaS. Components that are grouped together will be placed onto the same server such that the new group and placement can minimize the total resources allocated to the SaaS as well as the resource costs while satisfying the SaaS constraints. The proposed solution will try to achieve this aim with a minimum number of changes to the current placement. These will be incorporated in the objective function of the problem. There are two parts of the objective function that will be used as a basis to evaluate each of the solutions.

1) The cost of VMs used by the SaaS: VMs have their costs which is based on their resources' capacity, the higher the capacity, the more it will cost. To calculate the total cost of the virtual machines for a chromosome, the total VM cost to host the SaaS components, TC, will be the basis of the evaluation. This is defined as:

$$TC = \sum_{vm_{x,y} \in VM} Cost_{vm_{x,y}} \tag{7}$$

where

$$Cost_{vm_{x,y}} = \begin{cases} C_{vm_{x,y}}, & \exists_{vm_{x,y}} \mid P(ac_{i,j}) = vm_{x,y} \\ 0 & otherwise \end{cases} \tag{8}$$

The following equation is to normalize TC, and to ensure TC is less than the current placement cost:

$$F(TC) = \begin{cases} 0, & TC \geq initialCost \\ \frac{initialCost - TC}{initialCost}, & otherwise \end{cases} \tag{9}$$

2) The changes cost for a solution: Changing the current placement of a component from one VM to another requires some memory and bandwidth on both the source and destination servers. Greater resources will be needed for large components or a component with a large memory requirement. These will incur some costs. To estimate this cost, the calculation for placement changes are based on the size of the components as well as its memory requirement. The migration cost for all the SaaS, MC, will be based on the size of the component, $Sz_{ac_{i,j}}$ as well as its memory requirement, $M_{ac_{i,j}}$ which is defined below:

$$MC = \sum_{ac_{i,j} \in AC} \frac{Sz_{ac_{i,j}}}{max(Sz_{AC}) \times 2} + \frac{M_{ac_{i,j}}}{max(M_{AC}) \times 2} \tag{10}$$

The following equation is to normalize MC:

$$F(MC) = 1 - \frac{M}{N(AC)} \tag{11}$$

Based on the attributes that have been defined above, the fitness function for the algorithm is:

$$F(X) = (F(TC) \times w_1) + (F(MC) \times w_2) \tag{12}$$

where w_1 and w_2 are the weightage for each part and $w_1 + w_2 = 1$.

E. The Algorithm

In the beginning, the initial population is initialized randomly. A repairing function is imposed to repair any individual that violates the SaaS resource requirements and SaaS constraints. The fitness evaluation is done in two parts: the calculation of the VM cost and the calculation of the migration costs. The population then undergoes the genetic operations and fitter individuals will be copied to the next generation. These processes will be conducted iteratively until the termination condition is met. The following is the algorithm for the GGA.

Algorithm 1: Grouping Genetic Algorithm

```
1  bestFitness = 0
2  randomly initiliase (Population)
3  while termination condition is not true do
4      for X ∈ Population do
5          if X violates SaaS resource requirements, SaaS
           placement constraint or SaaS response time
           constraint then
6              Repair(X)
7          end
8          Calculate the new VM's cost
9          Calculate the cost of changing placement based
           on sequence of migration constraint
10         Calculate X fitness value, F(X)
11         if F(X) > bestFitness then
12             Replace bestFitness and store X
13         end
14     end
15     Select individuals from the Population based on
       roulette wheel selection
16     Probabilistically apply the crossover operator to
       generate new individual
17     Probabilistically select individuals for mutation
18     Use the new individuals to replace the old individuals
       in the Population
19 end
20 output bestFitness
```

Table III
SETS AND ATTRIBUTES OF CLOUD RESOURCES

Parameter	Value/Condition
Population size	100
Initial population	Randomly generated solutions
Crossover probability	0.95
Mutation probability	0.05
Termination condition	No improvement for the best individual in 25 consecutive generations

V. EVALUATION

The GGA described above has been implemented using Microsoft .NET Visual Studio C++ 6.0. Two experiments were conducted, the first is to evaluate the quality of the solutions produced by the GGA and the second experiment is to study the scalability of the GGA. In both experiments, we tested the GGA for five test cases that represent five different Cloud data centre sizes, from 300 to 1500 VMs, with an increment of 300. The number of composite SaaS is fixed at three, with a total of 15 application components and 6 data components. The parameter settings for the GGA are listed in Table III. For the fitness function, w_1 was set to 0.6 while w_2 was set to 0.4. The experiments were carried out on a desktop computer with 3 GHz Intel Core 2 Duo CPU and 4GB RAM.

To evaluate the quality of solutions produced by the GGA in the first experiment, we developed a First Fit Decreasing (FFD) heuristic for comparison. In the FFD heuristic, the VMs and SaaS components are sorted in decreasing order based on the capacity or requirement, and the heuristic will migrate each component to the first available VM. If the solution violates the time constraint, a new VM will be selected randomly. Considering the nature of both techniques, each of the test cases was repeated 10 times. Table IV shows the statistics of the experimental results including the best, worst, average and standard deviation of the VM costs for the GGA and the FFD. Fig. 5 visualises the VM costs for the two techniques.

Based on the results, it can be seen that the GGA always produced solutions that have lower VM costs than the FFD with significant savings of around 20%-30%, hence a better reconfiguration placement plan for the composite SaaS. It should also be noted that all these solutions have a lower migration cost than the FFD solutions.

We also analyzed the performance differences between the GGA and the FFD. A series of one-tailed t-tests indicated the statistics are significantly diffrerent ($p < 0.01$). The solutions generated by the proposed GGA outperformed the FFD in all test cases.

For the second experiment, Fig. 6 visualises the average computation time taken by the GGA and the FFD for finding the solutions for each of the test cases. It shows that the computation time of the GGA grows closely to linear with the Cloud data centre size and the longest computation time is below two minutes. However, there is a big gap with

Table IV
EXPERIMENTAL RESULTS OF THE GGA AND THE FFD FOR ALL TEST CASES

	Problem Size				VM Costs (GGA)				VM Costs (FFD)			
Test Case ID	VM	S	AC	DC	Best	Worst	Ave	StDev	Best	Worst	Ave	StDev
1	300	3	15	6	27.4	31	29.3	1.4	40.1	43.2	41.7	1.3
2	600	3	15	6	26.7	32.6	29.6	1.7	39.3	42.6	41.5	0.8
3	900	3	15	6	26.5	32.3	28.6	1.5	38	41.1	40.2	1.5
4	1200	3	15	6	25.6	28.9	27	1	32.4	36.9	34.3	1.5
5	1500	3	15	6	26.2	29.5	28.4	1.6	32.3	37.1	35.2	1.9

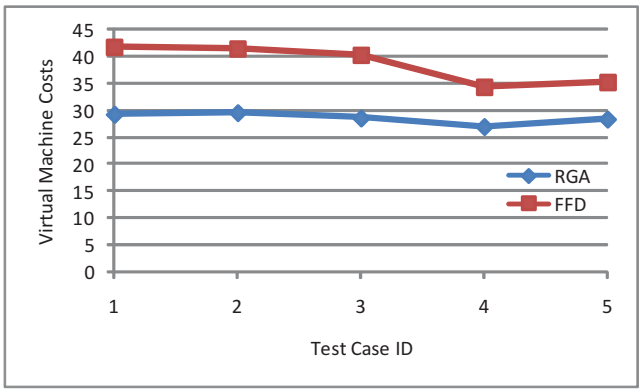

Figure 5. Comparison of the VMs costs produced by the GGA and FFD for all test cases

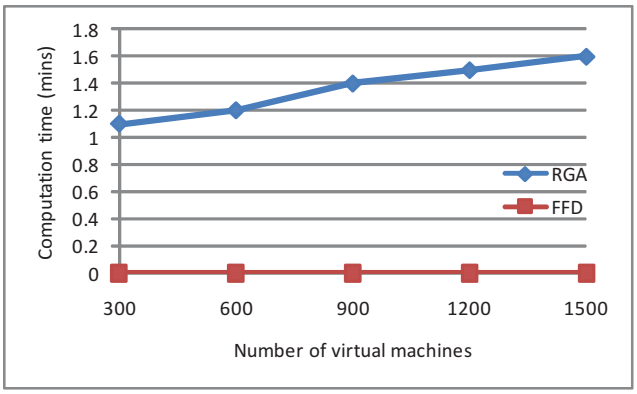

Figure 6. Computation times of the GGA and the FFD for different sizes of Cloud data centre

the computation time taken by the FFD which is less than one minute for each of the test cases. Although the time differences are significant, considering the large improvement in minimizing the resource usage by the GGA, this is still affordable. Furthermore, the maintenance phase of the SaaS reconfiguration placement in the Cloud occurs at different time scales, from seconds to days, depending on the data centre's needs.

VI. Conclusion and Future Work

We have presented the problem formulation and modeling of the multiple composite SaaS component clustering problem for the dynamic resource management of Cloud data centres. The major objective of the problem is to minimize the usage of resources of the SaaS without violating their SLAs by reconfiguring the placement of the applications' components. Meanwhile, it also aims to achieve the objective with the minimum changes possible.

A Grouping Genetic Algorithm (GGA) has been proposed and implemented. The GGA is specifically designed to cater for the structural group of a composite SaaS. The clustering and reconfiguration placement problem considers not only the resource requirements of the SaaS, but the communication needs of other application components, as well as the data components. To the best of our knowledge, this is the first attempt to handle the multiple composite SaaS reconfiguration placement in a dynamic Cloud environment. Based on the experimental results, the proposed GGA always produces a feasible solution for all test problems. It can also be seen that the new placement that was proposed by the GGA can save the resources consumed by the SaaS. Although the computation time taken is quite long, it is still acceptable considering that there are various types of maintenance in a data centre that are conducted at different time scales.

As for future work, we note that there is room for optimization in the implementation of the algorithm to improve its computation time.Although the algorithm is scalable, the computation time taken in finding the solutions can be further improved by implementing the GGA in a parallel manner. The network can be decomposed into several segments, and the solution can be executed in parallel based on the segmentations.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments on this paper.

This research was carried out as part of the activities of, and funded by the Smart Services Cooperative Research Centre (CRC) through the Australian Government's CRC Programme (Department of Innovation, Industry, Science and Research).

The study of Zeratul Izzah Mohd Yusoh was sponsored by the Ministry of Higher Education Malaysia through Universiti Teknikal Malaysia Melaka.

REFERENCES

[1] Foster, I., Yong, Z., Raicu, I., & Lu, S. (2008). Cloud Computing and Grid Computing 360-Degree Compared. In *Grid Computing Environments Workshop* (pp. 1-10). Austin, Texas: IEEE..

[2] Vaquero, L. M., Rodero-Merino, L., Caceres, J., & Lindner, M. (2009). A Break in the Clouds: Towards a Cloud Definition. *SIGCOMM Computer Communication Review*, 39(1), 50-55.

[3] Candan, K. S., Li, W.-S., Phan, T., & Zhou, M. (2009). Frontiers in information and Software as Services. In *Proceeding of the IEEE 25th International Conference on Data Engineering* (pp. 1761-1768). Shanghai, China: IEEE.

[4] Dubey, A., & Wagle, D. (2007). Delivering Software as a Service. *The McKinsey Quarterly*, 1-12.

[5] Cisco System Inc. (2008). Cisco Service-Oriented Network Architecture: Support and Optimize SOA and Web 2.0 Applications [Electronic Version]. Retrieved Mac 2011, from http://www.cisco.com/

[6] Jayasinghe, D., Pu, C., Eilam, T., Steinder, M., Whally, I., & Snible, E. (2011). Improving Performance and Availability of Services Hosted on IaaS Clouds with Structural Constraint-Aware Virtual Machine Placement. In *Proceeding of the IEEE International Conference on Services Computing* (pp. 72-79). Washington, USA: IEEE.

[7] Verma, A., Ahuja, P., & Neogi, A. (2008). pMapper: power and migration cost aware application placement in virtualized systems. In *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware* (pp. 243-264). New York, USA: Springer-Verlag New York, Inc.

[8] Wood, T., Shenoy, P., Venkataramani, A., & Yousif, M. (2007). Black-box and gray-box strategies for virtual machine migration. In *Proceeding of the 4th USENIX Symposium on Networked Systems Design & Implementation* (Vol. 7, pp. 229-242). Cambridge: USENIX.

[9] Hermenier, F., Lorca, X., Menaud, J. M., Muller, G., & Lawall, J. (2009). Entropy: a consolidation manager for clusters. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments* (pp. 41-50). New York, USA: ACM.

[10] Wang, Y., & Wang, X. (2010) Power optimization with performance assurance for multi-tier applications in virtualized data centers. In *Proceeding of the 2010 39th International Conference on Parallel Processing Workshops (ICPPW)*, (pp. 512-519). San Diego, CA: IEEE.

[11] Cucinotta, T., Palopoli, L., Abeni, L., Faggioli, D., & Lipari, G. (2010). On the Integration of Application Level and Resource Level QoS Control for Real-Time Applications. In *Proceeding of the IEEE Transactions on Industrial Informatics* (Vol. 6, pp. 479-491): IEEE.

[12] Xu, J., & Fortes, J. A. B. (2010). Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments. In *Proceeding of the 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, Green Computing and Communications (GreenCom) (pp. 179-188).

[13] Gao, Q., Tang, P., Deng, T., & Wo, T. (2011). VirtualRank: A Prediction Based Load Balancing Technique in Virtual Computing Environment. In *Proceeding of the 2011 IEEE World Congress on Services (SERVICES)* (pp. 247-256). Washington DC: IEEE.

[14] Andreolini, M., Casolari, S., Colajanni, M., & Messori, M. (2010). Dynamic load management of virtual machines in cloud architectures. *Cloud Computing*, 34, 201-214.

[15] Falkenauer, E. (1996). A hybrid grouping genetic algorithm for bin packing. *Journal of heuristics*, 2(1), 5-30.

[16] Mitchell, M. (1998). An introduction to genetic algorithms: The MIT press.

[17] Yusoh, Z., & Tang, M. (2010): A Penalty-based Genetic Algorithm for the Composite SaaS Placement Problem in the Cloud. In *IEEE World Congress on Computational Intelligence*. (pp. 600-607). IEEE: Spain